



SOFTWARE TOOL ARTICLE

TreeSummarizedExperiment: a S4 class for data with hierarchical structure [version 1; peer review: 2 approved, 1 approved with reservations]

Ruizhu Huang ^{1,2}, Charlotte Soneson ¹⁻³, Felix G.M. Ernst ⁴, Kevin C. Rue-Albrecht⁵, Guangchuang Yu ^{6,7}, Stephanie C. Hicks ⁸, Mark D. Robinson ^{1,2}

¹Department of Molecular Life Sciences, University of Zurich, Zurich, Switzerland

²SIB Swiss Institute of Bioinformatics, Zurich, Switzerland

³Friedrich Miescher Institute for Biomedical Research, Basel, Switzerland

⁴Population Health Sciences, German Center for Neurodegenerative Diseases (DZNE), Bonn, Germany

⁵MRC WIMM Centre for Computational Biology, University of Oxford, Oxford, OX3 9DS, UK

⁶Department of Bioinformatics, School of Basic Medical University, Guangzhou, Guangdong, China

⁷Microbiome Medicine Center, Division of Laboratory Medicine, Zhujiang Hospital, Southern Medical University, Guangzhou, Guangdong, China

⁸Department of Biostatistics, Johns Hopkins Bloomberg School of Public Health, Baltimore, USA

V1 First published: 15 Oct 2020, 9:1246
<https://doi.org/10.12688/f1000research.26669.1>
 Latest published: 02 Mar 2021, 9:1246
<https://doi.org/10.12688/f1000research.26669.2>

Abstract

Data organized into hierarchical structures (e.g., phylogenies or cell types) arises in several biological fields. It is therefore of interest to have data containers that store the hierarchical structure together with the biological profile data, and provide functions to easily access or manipulate data at different resolutions. Here, we present TreeSummarizedExperiment, a R/S4 class that extends the commonly used SingleCellExperiment class by incorporating tree representations of rows and/or columns (represented by objects of the phylo class). It follows the convention of the SummarizedExperiment class, while providing links between the assays and the nodes of a tree to allow data manipulation at arbitrary levels of the tree. The package is designed to be extensible, allowing new functions on the tree (phylo) to be contributed. As the work is based on the SingleCellExperiment class and the phylo class, both of which are popular classes used in many R packages, it is expected to be able to interact seamlessly with many other tools.

Keywords

SummarizedExperiment, tree, microbiome, hierarchical structure

Open Peer Review

Approval Status

	1	2	3
version 2			
(revision)			
02 Mar 2021			
version 1			
15 Oct 2020			

1. **Leo Lahti** , University of Turku, Turku, Finland
2. **Shila Ghazanfar** , University of Cambridge, Cambridge, UK
3. **Matthew E. Ritchie** , The Walter and Eliza Hall Institute of Medical Research, Parkville, Australia

Any reports and responses or comments on the

article can be found at the end of the article.



This article is included in the **Bioinformatics** gateway.



This article is included in the **Bioconductor** gateway.



This article is included in the **RPackage** gateway.

Corresponding author: Mark D. Robinson (mark.robinson@imls.uzh.ch)

Author roles: **Huang R:** Conceptualization, Software, Writing – Original Draft Preparation, Writing – Review & Editing; **Soneson C:** Conceptualization, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing; **Ernst FGM:** Software, Writing – Review & Editing; **Rue-Albrecht KC:** Conceptualization, Writing – Review & Editing; **Yu G:** Conceptualization, Writing – Review & Editing; **Hicks SC:** Conceptualization, Writing – Review & Editing; **Robinson MD:** Conceptualization, Supervision, Writing – Original Draft Preparation, Writing – Review & Editing

Competing interests: No competing interests were disclosed.

Grant information: This work was supported by the Swiss National Science Foundation (grant number 310030_175841). MDR acknowledges support from the University Research Priority Program Evolution in Action at the University of Zurich. SCH is supported by CZF2019-002443 from the Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation.

Copyright: © 2020 Huang R *et al.* This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

How to cite this article: Huang R, Soneson C, Ernst FGM *et al.* **TreeSummarizedExperiment: a S4 class for data with hierarchical structure [version 1; peer review: 2 approved, 1 approved with reservations]** F1000Research 2020, 9:1246 <https://doi.org/10.12688/f1000research.26669.1>

First published: 15 Oct 2020, 9:1246 <https://doi.org/10.12688/f1000research.26669.1>

Introduction

Biological data arranged into a hierarchy occurs in several fields. A notable example is in microbial survey studies, where the microbiome is profiled with amplicon sequencing or whole genome shotgun sequencing, and microbial taxa are organized as a tree according to their similarities in the genomic sequence or the evolutionary history. Also, a tree might be used in single cell cytometry or RNA-seq data, with nodes representing cell sub-populations at different granularities¹. Currently, *phyloseq*² and *SingleCellExperiment*³ are popular classes used in the analysis of microbial data and single cell data, respectively. The former supports the information pertaining to the hierarchical structure that is available as the *phylo* class (e.g., phylogenetic tree), and the latter is derived from the *SummarizedExperiment* class (defined in the *SummarizedExperiment* package⁴), which is widely used as a standardized container across many Bioconductor packages. Since the data structures in these fields share similarities, we were motivated to develop an S4 class⁵, *TreeSummarizedExperiment*, that not only leverages the facilities from the *SummarizedExperiment* class, but also bridges the functionality from the *phylo* class, which is available from the *ape*⁶ package and has been imported in more than 200 R packages.

We define *TreeSummarizedExperiment* by extending the *SingleCellExperiment* class, so that it is a member of the *SummarizedExperiment* family, and thus benefits from the comprehensive Bioconductor ecosystem (e.g., *iSEE*⁷, *SEtools*⁸, and *ggbio*⁹). At the same time, all slots of the *phyloseq* class have their corresponding slots in the *TreeSummarizedExperiment* class, which enables convenient conversion between these classes. Furthermore, we allow the link between profile data and nodes of the tree, including leaves and internal nodes, which is useful for algorithms in the downstream analysis that need to access internal nodes of the tree (e.g., *treeclimbR*¹).

Methods

R version: R version 4.0.2 (2020-06-22)

Bioconductor version: 3.11

Package: 1.4.8

Implementation

The structure of *TreeSummarizedExperiment*. The structure of the *TreeSummarizedExperiment* class is shown in Figure 1.

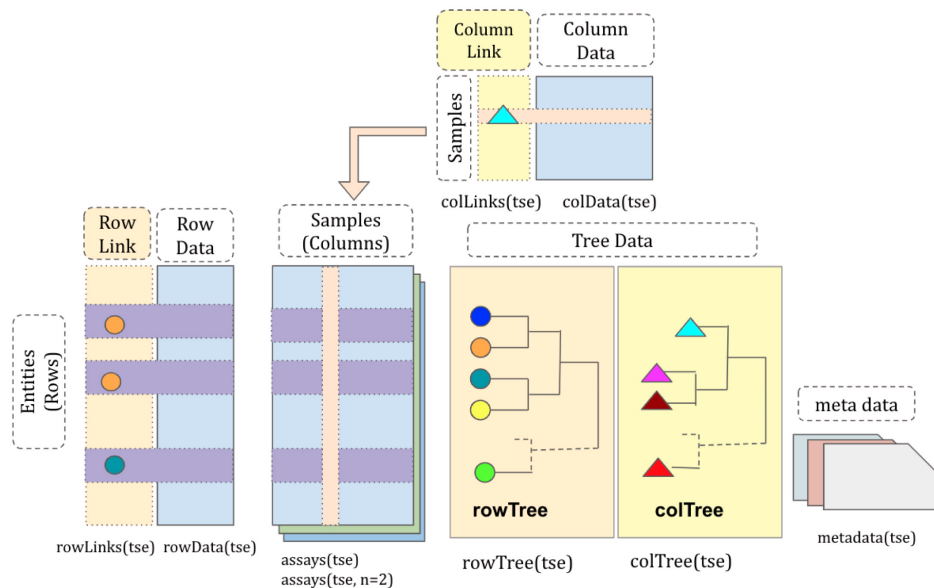


Figure 1. The structure of the *TreeSummarizedExperiment* class. The rectangular data matrices are stored in assays. Each matrix usually has rows representing entities (e.g., genes or microbial taxa) and columns representing cells or samples. Information about rows and columns is stored in *rowData* and *colData*, respectively. The hierarchy structure on rows or columns is stored in *rowTree* or *colTree* respectively, and the link information between rows/columns and nodes of the row/column tree is in *rowLinks* *colLinks*.

Compared to the *SingleCellExperiment* objects, *TreeSummarizedExperiment* has four additional slots:

- `rowTree`: the hierarchical structure on the rows of the assays.
- `rowLinks`: the link information between rows of the assays and the `rowTree`.
- `colTree`: the hierarchical structure on the columns of the assays.
- `colLinks`: the link information between columns of the assays and the `colTree`.

The `rowTree` and/or `colTree` can be left empty (NULL) if no trees are available; in this case, the `rowLinks` and/or `colLinks` are also set to NULL. All other *TreeSummarizedExperiment* slots are inherited from *SingleCellExperiment*.

The `rowTree` and `colTree` slots require the tree to be an object of the `phylo` class. If a tree is available in an alternative format, it can often be converted to a `phylo` object using dedicated R packages (e.g., *treeio*¹⁰).

Functions in the *TreeSummarizedExperiment* package fall in two main categories: operations on the *TreeSummarizedExperiment* object or operations on the tree (`phylo`) objects. The former includes constructors and accessors, and the latter serves as “components” to be assembled as accessors or functions that manipulate the *TreeSummarizedExperiment* object. Given that more than 200 R packages make use of the `phylo` class, there are many resources (e.g., *ape*) for users to manipulate the small “pieces” in addition to those provided in *TreeSummarizedExperiment*.

The toy datasets as the example data

We generate a toy dataset that has observations of 6 entities collected from 4 samples as an example to show how to construct a *TreeSummarizedExperiment* object.

```
library(TreeSummarizedExperiment)

# assays data (typically, representing observed data from an experiment)
assay_data <- rbind(rep(0, 4), matrix(1:20, nrow = 5))
colnames(assay_data) <- paste0("sample", 1:4)
rownames(assay_data) <- paste0("entity", seq_len(6))
assay_data
```

```
##           sample1 sample2 sample3 sample4
## entity1         0         0         0         0
## entity2         1         6        11        16
## entity3         2         7        12        17
## entity4         3         8        13        18
## entity5         4         9        14        19
## entity6         5        10        15        20
```

The information of entities and samples are given in the `row_data` and `col_data`, respectively.

```
# row data (feature annotations)
row_data <- data.frame(Kingdom = "A",
                       Phylum = rep(c("B1", "B2"), c(2, 4)),
                       Class = rep(c("C1", "C2", "C3"), each = 2),
                       OTU = paste0("D", 1:6),
                       row.names = rownames(assay_data),
                       stringsAsFactors = FALSE)
row_data
```

```
##           Kingdom Phylum Class OTU
## entity1      A       B1     C1  D1
## entity2      A       B1     C1  D2
## entity3      A       B2     C2  D3
## entity4      A       B2     C2  D4
## entity5      A       B2     C3  D5
## entity6      A       B2     C3  D6
```

```
# column data (sample annotations)
col_data <- data.frame(gg = c(1, 2, 3, 3),
                      group = rep(LETTERS[1:2], each = 2),
                      row.names = colnames(assay_data),
                      stringsAsFactors = FALSE)

col_data
```

```
##           gg group
## sample1    1     A
## sample2    2     A
## sample3    3     B
## sample4    3     B
```

The hierarchical structure of the 6 entities and 4 samples are denoted as **row_tree** and **col_tree**, respectively. The two trees are phylo objects randomly created with `rtree` from the package *ape*. Note that the row tree has 5 rather than 6 leaves; this is used later to show that multiple rows in the assays are allowed to map to a single node in the tree.

```
library(ape)

# The first toy tree
set.seed(12)
row_tree <- rtree(5)

# The second toy tree
set.seed(12)
col_tree <- rtree(4)

# change node labels
col_tree$tip.label <- colnames(assay_data)
col_tree$node.label <- c("All", "GroupA", "GroupB")
```

We visualize the tree using the package *ggtree* 2.2.4¹¹. Here, the internal nodes of the **row_tree** have no labels as shown in **Figure 2**.

```
library(ggtree)
library(ggplot2)

# Visualize the row tree
ggtree(row_tree, size = 2, branch.length = "none") +
  geom_text2(aes(label = node), color = "darkblue",
            hjust = -0.5, vjust = 0.7, size = 4) +
  geom_text2(aes(label = label), color = "darkorange",
            hjust = -0.1, vjust = -0.7, size = 4)
```

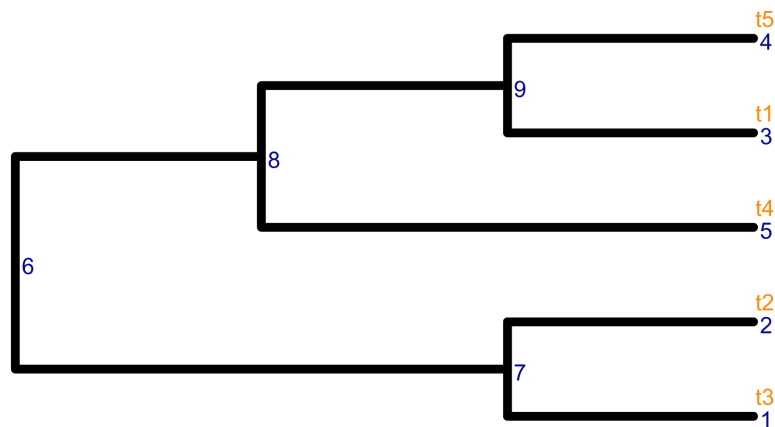


Figure 2. The structure of the row tree. The node labels and the node numbers are in orange and blue text, respectively.

The `col_tree` has labels for internal nodes as shown in **Figure 3**.

```
# Visualize the column tree
ggtree(col_tree, size = 2, branch.length = "none") +
  geom_text2(aes(label = node), color = "darkblue",
    hjust = -0.5, vjust = 0.7, size = 4) +
  geom_text2(aes(label = label), color = "darkorange",
    hjust = -0.1, vjust = -0.7, size = 4) +
  ylim(c(0.8, 4.5)) +
  xlim(c(0, 2.2))
```

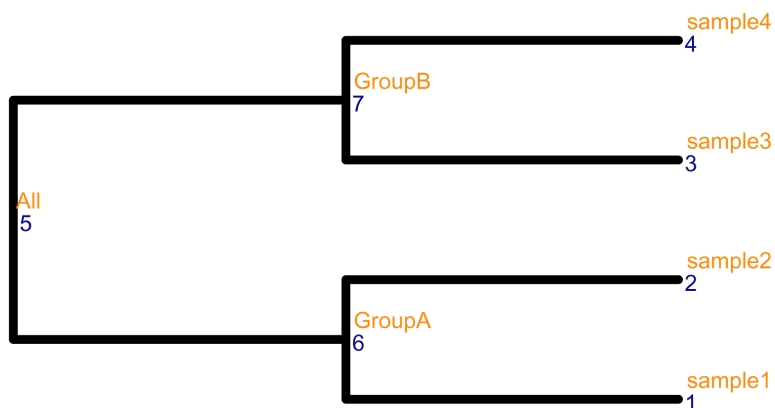


Figure 3. The structure of the column tree. The node labels and the node numbers are in orange and blue text, respectively.

The construction of *TreeSummarizedExperiment*

The *TreeSummarizedExperiment* class is used to store the toy data generated in the previous section: `assay_data`, `row_data`, `col_data`, `col_tree` and `row_tree`. To correctly store data, the link information between the rows (or columns) of `assay_data` and the nodes of the `row_tree` (or `col_tree`) can be provided via a character vector `rowNodeLab` (or `colNodeLab`), with length equal to the number of rows (or columns)

of the assays; otherwise the row (or column) names are used. Tree data takes precedence to determine entities included during the creation of the *TreeSummarizedExperiment* object; columns and rows with labels that are not present among the node labels of the tree are removed with warnings. The link data between the assays tables and the tree data is automatically generated during the construction.

The row and column trees can be included simultaneously during the construction of a *TreeSummarizedExperiment* object. Here, the column names of **assay_data** can be found in the node labels of the column tree, which enables the link to be created between the column dimension of **assay_data** and the column tree **col_tree**. If the row names of **assay_data** are not in the node labels of **row_tree**, we would need to provide their corresponding node labels (**row_lab**) to **rowNodeLab** in the construction of the object. It is possible to map multiple rows or columns to a node, for example, the same leaf label is used for the first two rows in **row_lab**.

```
# all column names could be found in the node labels of the column tree
all(colnames(assay_data) %in% c(col_tree$tip.label, col_tree$node.label))
```

```
## [1] TRUE
```

```
# provide the node labels in rowNodeLab
tip_lab <- row_tree$tip.label
row_lab <- tip_lab[c(1, 1:5)]
row_lab
```

```
## [1] "t3" "t3" "t2" "t1" "t5" "t4"
```

```
both_tse <- TreeSummarizedExperiment(assays = list(Count = assay_data),
                                     rowData = row_data,
                                     colData = col_data,
                                     rowTree = row_tree,
                                     rowNodeLab = row_lab,
                                     colTree = col_tree)
```

```
both_tse
```

```
## class: TreeSummarizedExperiment
## dim: 6 4
## metadata(0):
## assays(1): Count
## rownames(6): entity1 entity2 ... entity5 entity6
## rowData names(4): Kingdom Phylum Class OTU
## colnames(4): sample1 sample2 sample3 sample4
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (6 rows)
## rowTree: a phylo (5 leaves)
## colLinks: a LinkDataFrame (4 rows)
## colTree: a phylo (4 leaves)
```

When printed on screen, *TreeSummarizedExperiment* objects display information as the parent *SingleCell-Experiment* class followed by four additional lines for **rowLinks**, **rowTree**, **colLinks** and **colTree**.

The accessor functions

Slots inherited from the *SummarizedExperiment* class can be accessed in the traditional way (e.g., `assays()`, `rowData()`, `colData()` and `metadata()`). These functions are both getters and setters. To clarify, getters and setters are functions for users to retrieve and to overwrite data from the corresponding slots, respectively.

For new slots, we provide `rowTree` (and `colTree`) accessors to retrieve the row (column) trees, and `rowLinks` (and `colLinks`) to retrieve the link information between assays and nodes of the row (column) tree. Currently, these functions are getters but not setters. If the tree is not available, the corresponding link data is `NULL`.

```
# access trees
rowTree(both_tse)
```

```
##
## Phylogenetic tree with 5 tips and 4 internal nodes.
##
## Tip labels:
## [1] "t3" "t2" "t1" "t5" "t4"
##
## Rooted; includes branch lengths.
```

```
colTree(both_tse)
```

```
##
## Phylogenetic tree with 4 tips and 3 internal nodes.
##
## Tip labels:
## [1] "sample1" "sample2" "sample3" "sample4"
## Node labels:
## [1] "All"      "GroupA" "GroupB"
##
## Rooted; includes branch lengths.
```

```
# access the link data
(r_link <- rowLinks(both_tse))
```

```
## LinkDataFrame with 6 rows and 4 columns
##           nodeLab nodeLab_alias  nodeNum  isLeaf
##           <character> <character> <integer> <logical>
## entity1      t3      alias_1      1      TRUE
## entity2      t3      alias_1      1      TRUE
## entity3      t2      alias_2      2      TRUE
## entity4      t1      alias_3      3      TRUE
## entity5      t5      alias_4      4      TRUE
## entity6      t4      alias_5      5      TRUE
```

```
(c_link <- colLinks(both_tse))
```

```
## LinkDataFrame with 4 rows and 4 columns
##           nodeLab nodeLab_alias  nodeNum  isLeaf
##           <character> <character> <integer> <logical>
## sample1    sample1    alias_1      1      TRUE
## sample2    sample2    alias_2      2      TRUE
## sample3    sample3    alias_3      3      TRUE
## sample4    sample4    alias_4      4      TRUE
```

The link data objects are of the `LinkDataFrame` class, which extends the `DataFrame` class with the restriction that it has at least four columns:

- `nodeLab`: the labels of nodes on the tree
- `nodeLab_alias`: the alias labels of nodes on the tree

- `nodeNum`: the numbers of nodes on the tree
- `isLeaf`: whether the node is a leaf node

More details about the `DataFrame` class could be found in the *S4Vectors* R/Bioconductor package.

The subsetting function

A *TreeSummarizedExperiment* object can be subset in two different ways: `[]` to subset by rows or columns, and `subsetByNode` to retrieve row and/or columns that correspond to nodes of a tree. To preserve the original clustering information, the `rowTree` and `colTree` are kept identical after subsetting, while `rowLinks` and `rowData` are updated accordingly.

```
sub_tse <- both_tse[1:2, 1]
sub_tse
```

```
## class: TreeSummarizedExperiment
## dim: 2 1
## metadata(0):
## assays(1): Count
## rownames(2): entity1 entity2
## rowData names(4): Kingdom Phylum Class OTU
## colnames(1): sample1
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (2 rows)
## rowTree: a phylo (5 leaves)
## colLinks: a LinkDataFrame (1 rows)
## colTree: a phylo (4 leaves)
```

```
# the row data
rowData(sub_tse)
```

```
## DataFrame with 2 rows and 4 columns
##           Kingdom      Phylum      Class      OTU
##           <character> <character> <character> <character>
## entity1         A         B1         C1         D1
## entity2         A         B1         C1         D2
```

```
# the row link data
rowLinks(sub_tse)
```

```
## LinkDataFrame with 2 rows and 4 columns
##           nodeLab nodeLab_alias nodeNum  isLeaf
##           <character> <character> <integer> <logical>
## entity1         t3         alias_1         1      TRUE
## entity2         t3         alias_1         1      TRUE
```

```
# The first four columns are from colLinks data and the others from colData
cbind(colLinks(sub_tse), colData(sub_tse))
```

```
## DataFrame with 1 row and 6 columns
##           nodeLab nodeLab_alias nodeNum  isLeaf      gg
##           <character> <character> <integer> <logical> <numeric>
## sample1      sample1      alias_1         1      TRUE         1
##           group
##           <character>
## sample1      A
```

To subset by nodes, we specify the node by its node label or node number. Here, *entity1* and *entity2* are both mapped to the same node t3, so both of them are retained

```
node_tse <- subsetByNode(x = both_tse, rowNode = "t3")
rowLinks(node_tse)
```

```
## LinkDataFrame with 2 rows and 4 columns
##           nodeLab nodeLab_alias  nodeNum  isLeaf
##           <character> <character> <integer> <logical>
## entity1      t3      alias_1         1      TRUE
## entity2      t3      alias_1         1      TRUE
```

Subsetting simultaneously in both dimensions is also allowed.

```
node_tse <- subsetByNode(x = both_tse, rowNode = "t3",
                        colNode = c("sample1", "sample2"))
assays(node_tse)[[1]]
```

```
##           sample1 sample2
## entity1         0       0
## entity2         1       6
```

Changing the tree

The current tree can be replaced by a new one using `changeTree`. If the hierarchical information is available as a data.frame with each column representing a taxonomic level (e.g., *rowData*), we provide `toTree` to convert it into a phylo object that is further visualized in [Figure 4](#).

```
# The toy taxonomic table
(taxa <- rowData(both_tse))
```

```
## DataFrame with 6 rows and 4 columns
##           Kingdom  Phylum  Class  OTU
##           <character> <character> <character> <character>
## entity1         A        B1      C1     D1
## entity2         A        B1      C1     D2
## entity3         A        B2      C2     D3
## entity4         A        B2      C2     D4
## entity5         A        B2      C3     D5
## entity6         A        B2      C3     D6
```

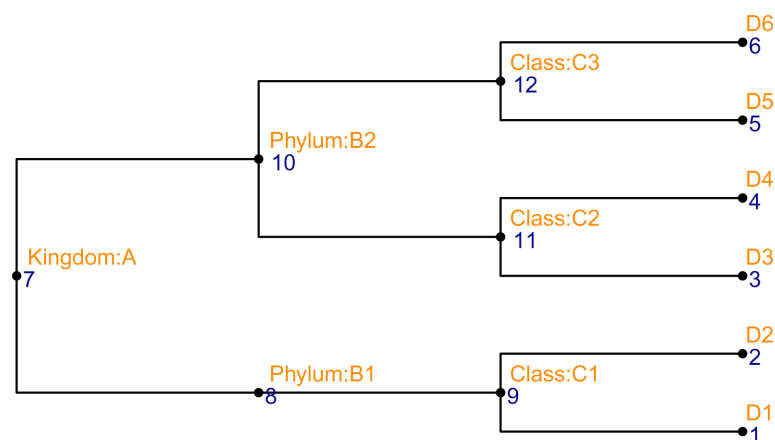


Figure 4. The structure of the taxonomic tree that is generated from the taxonomic table.

```
# convert it to a phylo tree
taxa_tree <- toTree(data = taxa)

# Viz the new tree
ggtree(taxa_tree) +
  geom_text2(aes(label = node), color = "darkblue",
             hjust = -0.5, vjust = 0.7, size = 4) +
  geom_text2(aes(label = label), color = "darkorange",
             hjust = -0.1, vjust = -0.7, size = 4) +
  geom_point2()
```

If the nodes of the two trees have a different set of labels, a vector mapping the nodes of the new tree must be provided in `rowNodeLab`.

```
taxa_tse <- changeTree(x = both_tse, rowTree = taxa_tree,
                      rowNodeLab = taxa[["OTU"]])
```

```
taxa_tse
```

```
## class: TreeSummarizedExperiment
## dim: 6 4
## metadata(0):
## assays(1): Count
## rownames(6): entity1 entity2 ... entity5 entity6
## rowData names(4): Kingdom Phylum Class OTU
## colnames(4): sample1 sample2 sample3 sample4
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (6 rows)
## rowTree: a phylo (6 leaves)
## colLinks: a LinkDataFrame (4 rows)
```

```
rowLinks(taxa_tse)
```

```
## LinkDataFrame with 6 rows and 4 columns
##           nodeLab nodeLab_alias  nodeNum  isLeaf
##      <character> <character> <integer> <logical>
## entity1         D1      alias_1         1     TRUE
## entity2         D2      alias_2         2     TRUE
## entity3         D3      alias_3         3     TRUE
## entity4         D4      alias_4         4     TRUE
## entity5         D5      alias_5         5     TRUE
## entity6         D6      alias_6         6     TRUE
```

Aggregation

Since it may be of interest to report or analyze observed data at multiple resolutions based on the provided tree(s), the *TreeSummarizedExperiment* package offers functionality to flexibly aggregate data to arbitrary levels of a tree.

The column dimension. Here, we demonstrate the aggregation functionality along the column dimension. The desired aggregation level is given in the `colLevel` argument, which can be specified using node labels (orange texts in Figure 3) or node numbers (blue texts in Figure 3). Furthermore, the summarization method used to aggregate multiple values can be specified via the argument `FUN`.

```
# use node labels to specify colLevel
agg_col <- aggValue(x = taxa_tse,
                   colLevel = c("GroupA", "GroupB"),
                   FUN = sum)
# or use node numbers to specify colLevel
agg_col <- aggValue(x = taxa_tse, colLevel = c(6, 7), FUN = sum)
```

```
assays(agg_col)[[1]]
```

```
##          alias_6 alias_7
## entity1         0      0
## entity2         7     27
## entity3         9     29
## entity4        11     31
## entity5        13     33
## entity6        15     35
```

The rowData does not change, but the colData is updated to reflect the metadata information that remains valid for the individual nodes after aggregation. For example, the column **group** has the A value for GroupA because the descendant nodes of GroupA all have the value A; whereas the column **gg** has the NA value for GroupA because the descendant nodes of GroupA have different values, (1 and 2).

```
# before aggregation
colData(taxa_tse)
```

```
## DataFrame with 4 rows and 2 columns
##          gg      group
##      <numeric> <character>
## sample1         1      A
## sample2         2      A
## sample3         3      B
## sample4         3      B
```

```
# after aggregation
colData(agg_col)
```

```
## DataFrame with 2 rows and 2 columns
##          gg      group
##      <numeric> <character>
## alias_6      NA      A
## alias_7       3      B
```

The colLinks is also updated to link the new rows of assays tables to the corresponding nodes of the column tree (Figure 3).

```
# the link data is updated
colLinks(agg_col)
```

```
## LinkDataFrame with 2 rows and 4 columns
##          nodeLab nodeLab_alias  nodeNum  isLeaf
##      <character> <character> <integer> <logical>
## alias_6      GroupA      alias_6         6    FALSE
## alias_7      GroupB      alias_7         7    FALSE
```

The row dimension. Similarly, we can aggregate rows to phyla by providing the names of the internal nodes that represent the phylum level (see taxa_one below).

```
# the phylum level
taxa <- c(taxa_tree$tip.label, taxa_tree$node.label)
(taxa_one <- taxa[startsWith(taxa, "Phylum:"))

## [1] "Phylum:B1" "Phylum:B2"

# aggregation
agg_taxa <- aggValue(x = taxa_tse, rowLevel = taxa_one, FUN = sum)
assays(agg_taxa)[[1]]

##           sample1 sample2 sample3 sample4
## alias_8         1         6         11         16
## alias_10        14        34         54         74
```

Users are nonetheless free to choose nodes from different taxonomic ranks for each final aggregated row. Note that it is not necessary to use all original rows during the aggregation process. Similarly, it is entirely possible for a row to contribute to multiple aggregated rows.

```
# A mixed level
taxa_mix <- c("Class:C3", "Phylum:B1")
agg_any <- aggValue(x = taxa_tse, rowLevel = taxa_mix, FUN = sum)
rowData(agg_any)

## DataFrame with 2 rows and 4 columns
##           Kingdom      Phylum      Class      OTU
##           <character> <character> <character> <logical>
## alias_12           A           B2           C3           NA
## alias_8             A           B1           C1           NA
```

Both dimensions. The aggregation on both dimensions could be performed in one step using the same function specified via FUN, in which case the `aggValue` function aggregates rows first, and columns second. If different functions are required for different dimensions, or if columns need to be aggregated before rows, users should perform the aggregation in two steps.

```
agg_both <- aggValue(x = both_tse, colLevel = c(6, 7),
                    rowLevel = 7:9, FUN = sum)
```

As expected, we obtain a table with 3 rows representing the aggregated row nodes 7, 8 and 9 (`rowLevel = 7:9`) and 2 columns representing the aggregated column nodes 6 and 7 (`colLevel = c(6, 7)`).

```
assays(agg_both)[[1]]
```

```
##           alias_6 alias_7
## alias_7         16      56
## alias_8         39      99
## alias_9         24      64
```

Functions operating on the phylo object.

Next, we highlight some functions to manipulate and/or to extract information from the `phylo` object. Further operations can be found in other packages, such as *ape*⁶, *tidytree*¹². These functions are useful for users who wish to develop more functions for the *TreeSummarizedExperiment* class.

To show these functions, we use the tree shown in [Figure 5](#).

```
data("tinyTree")
ggtree(tinyTree, branch.length = "none") +
  geom_text2(aes(label = label), hjust = -0.1, size = 3) +
  geom_text2(aes(label = node), vjust = -0.8,
            hjust = -0.2, color = "blue", size = 3)
```

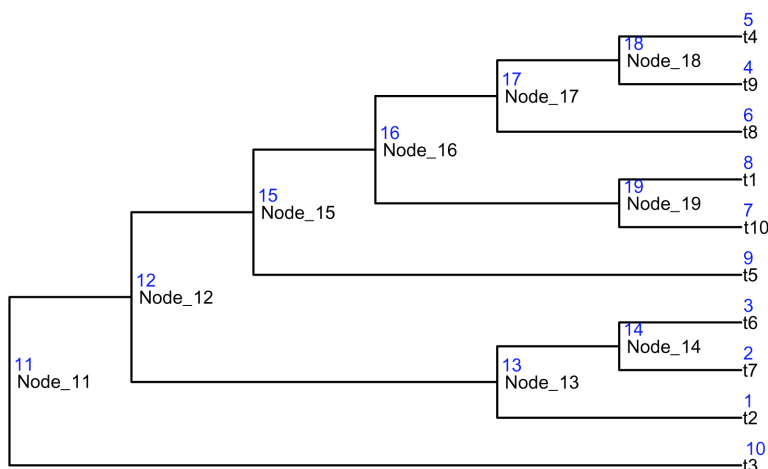


Figure 5. An example tree with node labels and numbers in black and blue texts, respectively.

Conversion of the node label and the node number The translation between the node labels and node numbers can be achieved by the function `convertNode`.

```
convertNode(tree = tinyTree, node = c(12, 1, 4))  
## [1] "Node_12" "t2"      "t9"  
  
convertNode(tree = tinyTree, node = c("t4", "Node_18"))  
##      t4 Node_18  
##      5      18
```

Find the descendants To get descendants that are at the leaf level, we could set the argument `only.leaf = TRUE` for the function `findDescendant`.

```
# only the leaf nodes
findDescendant(tree = tinyTree, node = 17, only.leaf = TRUE)

## $Node_17
## [1] 4 5 6
```

When `only.leaf = FALSE`, all descendants are returned.

```
# all descendant nodes
findDescendant(tree = tinyTree, node = 17, only.leaf = FALSE)

## $Node_17
## [1] 4 5 6 18
```

More functions. We list some functions that might also be useful in [Table 1](#). More functions are available in the package, and we encourage users to develop and contribute their own functions to the package.

Custom functions for the *TreeSummarizedExperiment* class

Here, we show examples of how to write custom functions for *TreeSummarizedExperiment* objects. To extract data corresponding to specific leaves, we created a function `subsetByLeaf` by combining functions working on the phylo class (e.g., `convertNode`, `keep.tip`, `trackNode`) with the accessor function `subsetByNode`. Here, `convertNode` and `trackNode` are available in *TreeSummarizedExperiment*, and `keep.tip` is from the *ape* package. Since the numeric identifier of a node is changed after pruning a tree

Table 1. A table lists some functions operating on the `phylo` object that are available in the *TreeSummarizedExperiment*.

Functions	Goal
<code>printNode</code>	print out the information of nodes
<code>countNode</code>	count the number of nodes
<code>distNode</code>	give the distance between a pair of nodes
<code>matTree</code>	list paths of a tree
<code>findAncestor</code>	find ancestor nodes
<code>findChild</code>	find child nodes
<code>findSibling</code>	find sibling nodes
<code>shareNode</code>	find the first node shared in the paths of nodes to the root
<code>unionLeaf</code>	find the union of descendant leaves
<code>trackNode</code>	track nodes by adding alias labels to a phylo object
<code>isLeaf</code>	test whether a node is a leaf node

with `keep.tip`, `trackNode` is provided to track the node and further update links between the rectangular assay matrices and the new tree.

```
# tse: a TreeSummarizedExperiment object
# rowLeaf: specific leaves
subsetByLeaf <- function(tse, rowLeaf) {
  # if rowLeaf is provided as node labels, convert them to node numbers
  if (is.character(rowLeaf)) {
    rowLeaf <- convertNode(tree = rowTree(tse), node = rowLeaf)
  }

  # subset data by leaves
  sse <- subsetByNode(tse, rowNode = rowLeaf)

  # update the row tree
  ## ----- new tree: drop leaves -----
  oldTree <- rowTree(sse)
  newTree <- ape::keep.tip(phy = oldTree, tip = rowLeaf)

  ## ----- update the row tree -----
  # track the tree
  track <- trackNode(oldTree)
  track <- ape::keep.tip(phy = track, tip = rowLeaf)

  # update the row tree:
  # 1. get the old alias label and update it to the new alias label
  # 2. provide the new alias label as rowNodeLab to update the row tree
  oldAlias <- rowLinks(sse)$nodeLab_alias
  newNode <- convertNode(tree = track, node = oldAlias)
  newAlias <- convertNode(tree = track, node = newNode)

  changeTree(x = sse, rowTree = newTree, rowNodeLab = newAlias)
}
```

The row tree is updated; after subsetting, it has only two leaves, `t2` and `t3`.

```
(both_sse <- subsetByLeaf(tse = both_tse, rowLeaf = c("t2", "t3")))
```

```
## class: TreeSummarizedExperiment
## dim: 3 4
## metadata(0):
## assays(1): Count
## rownames(3): entity1 entity2 entity3
## rowData names(4): Kingdom Phylum Class OTU
## colnames(4): sample1 sample2 sample3 sample4
## colData names(2): gg group
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (3 rows)
## rowTree: a phylo (2 leaves)
## colLinks: a LinkDataFrame (4 rows)
## colTree: a phylo (4 leaves)
```

```
rowLinks(both_sse)
```

```
## LinkDataFrame with 3 rows and 4 columns
##           nodeLab nodeLab_alias   nodeNum   isLeaf
##      <character> <character> <integer> <logical>
## entity1         t3      alias_1         1      TRUE
## entity2         t3      alias_1         1      TRUE
## entity3         t2      alias_2         2      TRUE
```

Operation

The *TreeSummarizedExperiment* package can be installed by following the standard installation procedures of Bioconductor packages.

```
# install BiocManager
if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
# install TreeSummarizedExperiment package
BiocManager::install("TreeSummarizedExperiment")
```

Minimum system requirements is R version 3.6 (or later) on a Mac, Windows or Linux system. We highly recommend to use the latest versions of R (currently, 4.0.2) and Bioconductor (at time of writing, 3.11) to gain access to the latest version of this package.

Use cases

To demonstrate the functionality of *TreeSummarizedExperiment*, we use it to store and manipulate a microbial dataset. We further show exploratory graphics using the available functions designed for *SummarizedExperiment* objects in other packages (e.g., *scater*), or customized functions from popular visualization packages (e.g., *ggplot2*¹³).

```
# Packages providing dataset
library(HMP16SData)

# Packages to manipulate data extracted from TreeSummarizedExperiment
library(tidyr)
library(dplyr)

# Packages providing visualization
library(ggplot2)
library(scales)
library(ggtree)
library(scater)
library(cowplot)
```

The Human Microbiome Project (HMP) 16S rRNA sequencing data, v35, is downloaded using the R package *HMP16SData*¹⁴, which contains survey data of samples collected at five major body sites in the variable regions 3–5. v35 is available as a SummarizedExperiment object via the ExperimentHub.

```
(v35 <- V35())
```

```
## class: SummarizedExperiment
## dim: 45383 4743
## metadata(2): experimentData phylogeneticTree
## assays(1): 16SrRNA
## rownames(45383): OTU_97.1 OTU_97.10 ... OTU_97.9998 OTU_97.9999
## rowData names(7): CONSENSUS_LINEAGE SUPERKINGDOM ... FAMILY GENUS
## colnames(4743): 700013549 700014386 ... 700114717 700114750
## colData names(7): RSID VISITNO ... HMP_BODY_SUBSITE SRS_SAMPLE_ID
```

```
# name the assay
names(assays(v35)) <- "Count"
```

The storage of HMP 16S rRNA-seq data

We store the phylogenetic tree as the rowTree. Links between nodes of the tree and rows of assays are automatically generated in the construction of the *TreeSummarizedExperiment* object, and are stored as rowLinks. Rows of the assays matrices that do not have a match to nodes of the tree are removed with warnings.

```
(tse_phy <- TreeSummarizedExperiment(assays = assays(v35),
                                     rowData = rowData(v35),
                                     colData = colData(v35),
                                     rowTree = metadata(v35)$phylogeneticTree,
                                     metadata = metadata(v35)["experimentData"])
```

```
## Warning in .linkFun(tree = rowTree, sce = sce, nodeLab = rowNodeLab, onRow
= TRUE):
```

```
47 row(s) couldn't be matched to the tree and are/is removed.
```

```
## class: TreeSummarizedExperiment
## dim: 45336 4743
## metadata(1): experimentData
## assays(1): Count
## rownames(45336): OTU_97.1 OTU_97.10 ... OTU_97.9998 OTU_97.9999
## rowData names(7): CONSENSUS_LINEAGE SUPERKINGDOM ... FAMILY GENUS
## colnames(4743): 700013549 700014386 ... 700114717 700114750
## colData names(7): RSID VISITNO ... HMP_BODY_SUBSITE SRS_SAMPLE_ID
## reducedDimNames(0):
## altExpNames(0):
## rowLinks: a LinkDataFrame (45336 rows)
## rowTree: a phylo (45364 leaves)
## colLinks: NULL
## colTree: NULL
```

```
cD <- colData(tse_phy)
dim(table(cD$HMP_BODY_SITE, cD$RUN_CENTER))
```

```
## [1] 5 12
```

Exploratory graphics

Here, we show *TreeSummarizedExperiment* working seamlessly with *SEtools* (v. 1.2.0) to prepare data for the exploratory graphics. Since all operational taxonomic units (OTUs) in the sample belong to Bacteria in the SUPERKINGDOM level, we can calculate the sequencing depths by aggregating counts to the SUPERKINGDOM level. The resultant *TreeSummarizedExperiment* object **agg_total** is further converted into a data frame **df_total** with selected columns (HMP_BODY_SITE and RUN_CENTER) from the column data.

```
library(SETools)
agg_total <- aggSE(x = tse_phy, by = "SUPERKINGDOM",
                  assayFun = sum)

# The assays data and selected columns of the row/col data are merged into a
# data frame
df_total <- meltSE(agg_total, genes = rownames(agg_total),
                  colDat.columns = c("HMP_BODY_SITE", "RUN_CENTER"))

head(df_total)
```

```
##      feature      sample      HMP_BODY_SITE RUN_CENTER Count
## 1 Bacteria 700013549 Gastrointestinal Tract      BCM 5295
## 2 Bacteria 700014386 Gastrointestinal Tract    BCM,BI 10811
## 3 Bacteria 700014403              Oral      BCM,BI 12312
## 4 Bacteria 700014409              Oral    BCM,BI 20355
## 5 Bacteria 700014412              Oral    BCM,BI 14021
## 6 Bacteria 700014415              Oral    BCM,BI 17157
```

To make harmonized figures with **ggplot2** (v. 3.3.2)¹³, we customized a theme to be applied to several plots in this section.

```
# Customized the plot theme
prettify <- theme_bw(base_size = 10) + theme(
  panel.spacing = unit(0, "lines"),
  axis.text = element_text(color = "black"),
  axis.text.x = element_text(angle = 45, hjust = 1),
  legend.key.size = unit(6, "mm"),
  legend.spacing.x = unit(1, "mm"),
  plot.title = element_text(hjust = 0.5),
  legend.text = element_text(size = 9),
  legend.position = "bottom",
  strip.background = element_rect(colour = "black", fill = "gray90"),
  strip.text.x = element_text(color = "black", size = 10),
  strip.text.y = element_text(color = "black", size = 10))
```

From **Figure 6**, we note that more samples were collected from the oral site than other body sites.

```
# Figure: (the number of samples) VS (centers)
ggplot(df_total) +
  geom_bar(aes(RUN_CENTER, fill = HMP_BODY_SITE),
           position = position_dodge()) +
  labs(title = "The number of samples across centers", y = "") +
  scale_fill_brewer(palette = "Set1") +
  prettify
```

Figure 7 shows that the sequencing depth of each sample across different coordination centers are quite similar. Within the coordination center, samples collected from Skin are more variable in the sequencing depth than those from other body sites.

```
# Figure: (the sequencing depths) VS (centers)
ggplot(df_total) +
  geom_boxplot(aes(x = RUN_CENTER, y = Count, fill = HMP_BODY_SITE),
              position = position_dodge()) +
  labs(title = "The sequencing depths of samples") +
  scale_y_continuous(trans = log10_trans()) +
  scale_fill_brewer(palette = "Set1") +
  labs(y = "Total counts") +
  prettify
```

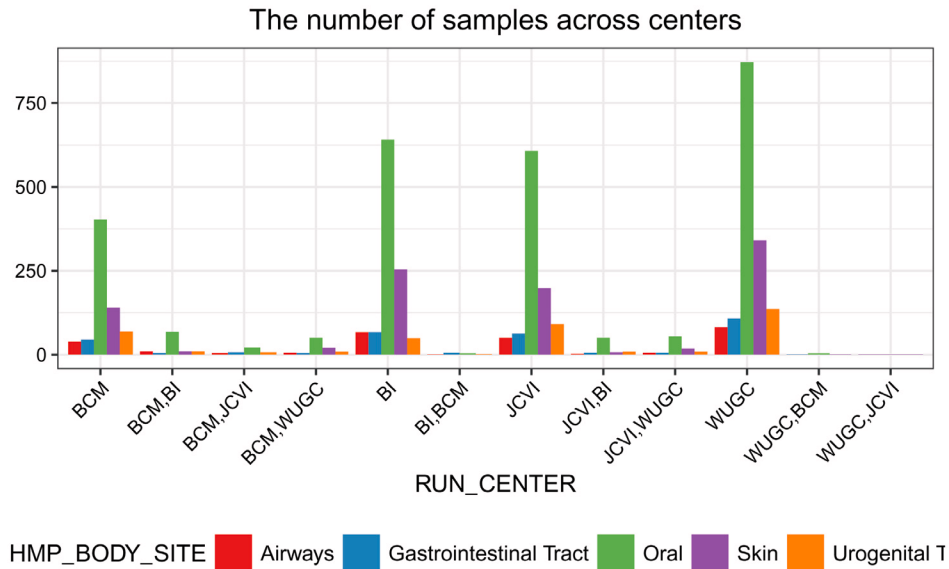


Figure 6. The number of samples from different research centers: Baylor College of Medicine (BCM), the Broad Institute (BI), the J. Craig Venter Institute (JCVI) and Washington University (WUGC). Samples collected at different body sites (HMP_BODY_SITE) are in different colors.

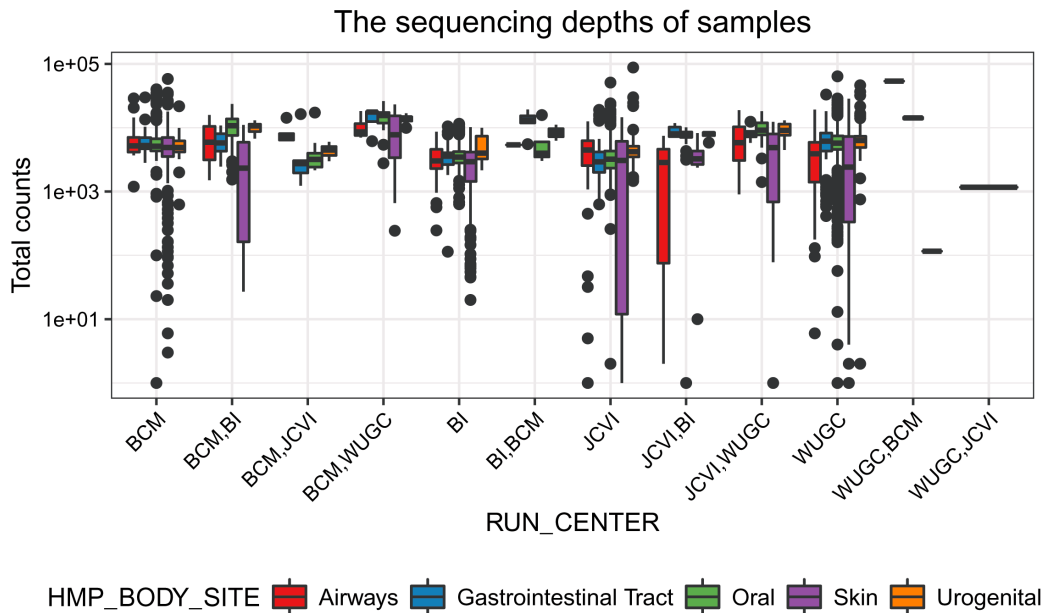


Figure 7. The sequencing depth of samples from different research centers. Samples collected at different body sites are in different colors.

Dimensionality reduction

We visualize samples in reduced dimensions to see whether those from the same body site are similar to each other. Three dimensionality reduction techniques are available in the package *scater* (v. 1.16.2), including principal component analysis (PCA)¹⁵, t-distributed Stochastic Neighbor Embedding (t-SNE)¹⁶, and uniform manifold approximation and projection (UMAP)¹⁷. Since *TreeSummarizedExperiment* extends the *SingleCellExperiment* class, functions from *scater*¹⁸ can be used directly. Here, we first apply PCA and t-SNE on data at the

operational taxonomic unit (OTU) level, and select the one better clustering the samples to apply on data aggregated at coarser taxonomic levels to see whether the resolution affects the separation of samples.

PCA and t-SNE at the OTU level

The PCA is performed on the log-transformed counts that are stored in the `assays` matrix with the name `logcounts`. In practice, data normalization is usually applied prior to the downstream analysis, to address bias or noise introduced during the sampling or sequencing process (e.g., uneven sampling depth). Here, the library size is highly variable (Figure 7) and non-zero OTUs vary across body sites. It is difficult to say what is the optimal normalization strategy, and the use of an inappropriate normalization method might introduce new biases. The discussion of normalization is outside the scope of this work. To keep it simple, we will visualize data without further normalization.

In Figure 8, we see that the Oral samples are distinct from those of other body sites. Samples from Skin, Urogenital Tract, Airways and Gastrointestinal Tract are not separated very well in the first two principal components.

```
# log-transformed data
assays(tse_phy)$logcounts <- log(assays(tse_phy)$Count + 1)

# run PCA at the OTU level
tse_phy <- runPCA(tse_phy, name="PCA_OTU", exprs_values = "logcounts")

# plot samples in the reduced dimensions
plotReducedDim(tse_phy, dimred = "PCA_OTU",
               colour_by = "HMP_BODY_SITE") +
  labs(title = "PCA at the OTU level") +
  guides(fill = guide_legend(override.aes = list(size=2.5, alpha = 1))) +
  theme(plot.title = element_text(hjust = 0.5),
        legend.position = "bottom")
```

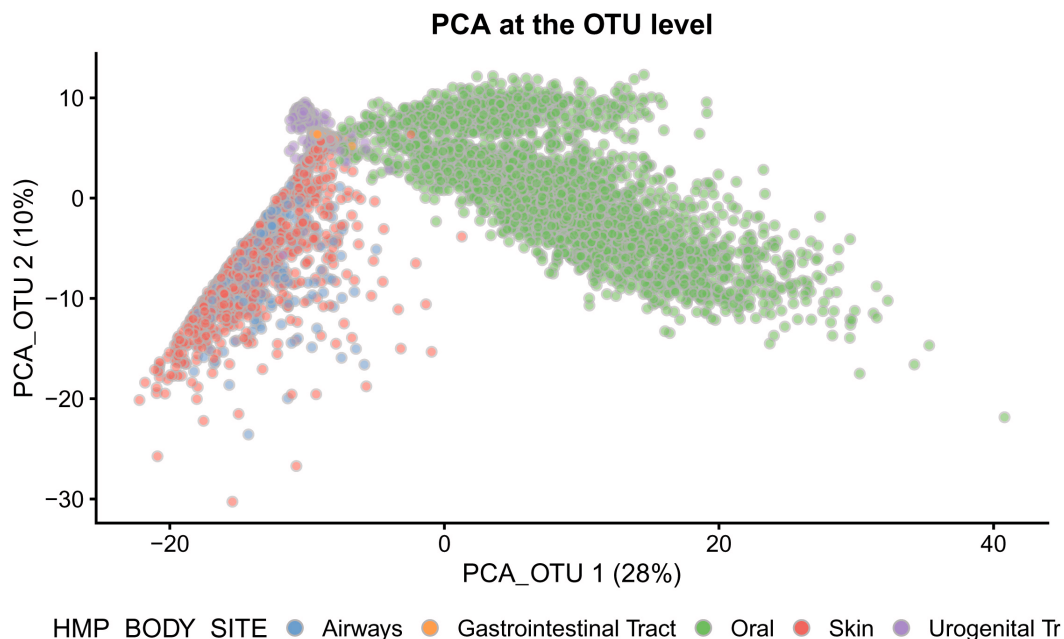


Figure 8. Principal component analysis (PCA) plot of samples using data at the OTU level. The first two principal components (PCs) are plotted. Each point represents a sample. Samples are coloured according to the body sites.

The separation is well improved with the use of t-SNE in [Figure 9](#). Samples from Oral, Gastrointestinal Tract, and Urogenital Tract form distinct clusters. Skin samples and airways samples still overlap.

```
# run t-SNE at the OTU level
tse_phy <- runTSNE(tse_phy, name="TSNE_OTU", exprs_values = "logcounts")

# plot samples in the reduced dimensions
tsne_otu <- plotReducedDim(tse_phy, dimred = "TSNE_OTU",
  colour_by = "HMP_BODY_SITE") +
  labs(title = "t-SNE at the OTU level") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_brewer(palette = "Set1") +
  labs(fill = "Body sites") +
  guides(fill = guide_legend(override.aes = list(size=2.5, alpha = 1))) +
  theme(plot.title = element_text(hjust = 0.5),
    legend.position = "bottom")
tsne_otu
```

Notably, there are two well-separated clusters labelled as oral samples. The smaller cluster includes samples from the Supragingival Plaque and Subgingival Plaque sites, while the larger cluster includes samples from other oral sub-sites ([Figure 10](#)).

```
is_oral <- colData(tse_phy)$HMP_BODY_SITE %in% "Oral"
colData(tse_phy)$from_plaque <- grepl(pattern = "Plaque",
  colData(tse_phy)$HMP_BODY_SUBSITE)

# Oral samples
plotReducedDim(tse_phy[, is_oral], dimred = "TSNE_OTU",
  colour_by = "from_plaque") +
  guides(fill = guide_legend(override.aes = list(size=2.5, alpha = 1))) +
  theme(plot.title = element_text(hjust = 0.5),
    legend.position = "bottom")
```

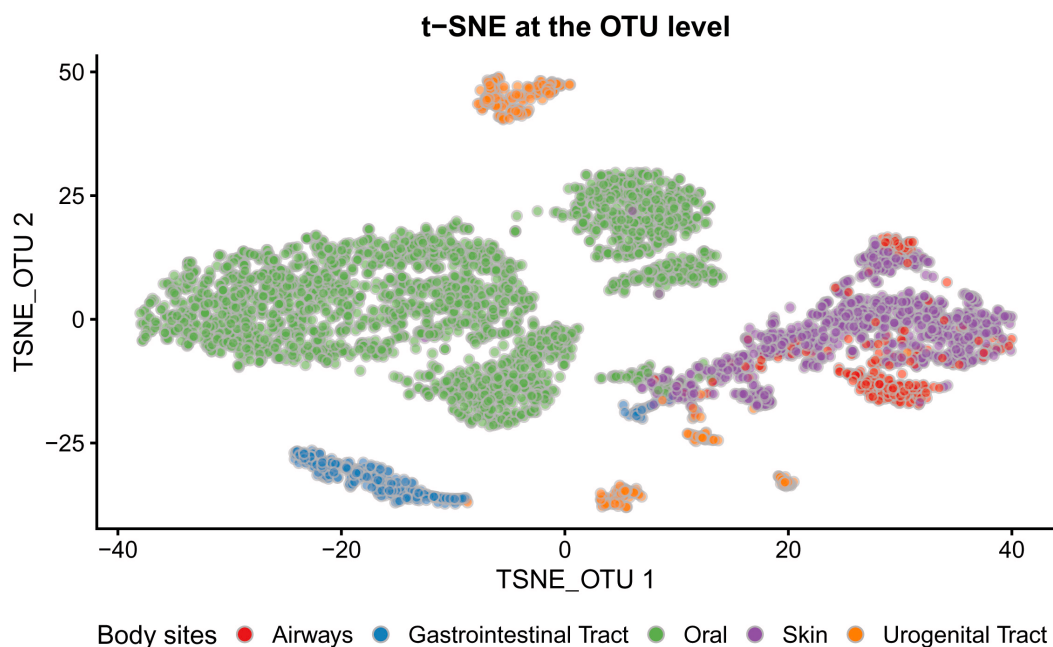


Figure 9. t-distributed Stochastic Neighbor Embedding (t-SNE) plot of samples using data at the OTU level. The first two t-SNE components are plotted. Each point represents a sample. Samples are coloured according to the body site.

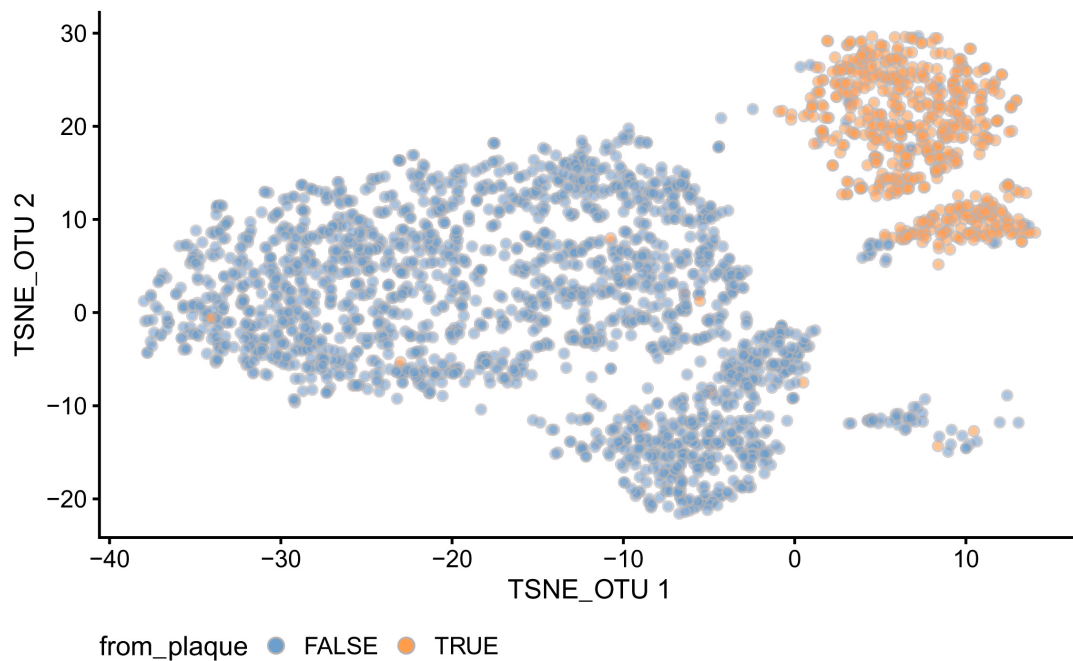


Figure 10. t-distributed Stochastic Neighbor Embedding (t-SNE) plot of samples from the oral site using data at the OTU level. The two t-SNE components computed are plotted. Each point is a sample. Samples from the 'supragingival or subgingival Plaque' are in orange, and those from other oral sub-sites are in blue.

t-SNE on broader taxonomic levels

To organize data at different taxonomic levels, we first replace the phylogenetic tree with the taxonomic tree that is generated from the taxonomic table. Due to the existence of polyphyletic groups, a tree structure cannot be generated. For example, the Alteromonadaceae family is from different orders: Alteromonadales and Oceanospirillales.

```
# taxonomic tree
tax_order <- c("SUPERKINGDOM", "PHYLUM", "CLASS",
              "ORDER", "FAMILY", "GENUS", "CONSENSUS_LINEAGE")
tax_0 <- data.frame(rowData(tse_phy)[, tax_order])
tax_loop <- detectLoop(tax_tab = tax_0)

# show loops that are not caused by NA
head(tax_loop[!is.na(tax_loop$child), ])
```

##	parent	child	parent_column	child_column
## 35	Alteromonadales	Alteromonadaceae	ORDER	FAMILY
## 36	Oceanospirillales	Alteromonadaceae	ORDER	FAMILY
## 37	Rhizobiales	Rhodobacteraceae	ORDER	FAMILY
## 38	Rhodobacterales	Rhodobacteraceae	ORDER	FAMILY
## 39	Chromatiales	Sinobacteraceae	ORDER	FAMILY
## 40	Xanthomonadales	Sinobacteraceae	ORDER	FAMILY

To resolve the loops, we add a suffix to the polyphyletic genus with `resolveLoop`. For example, *Ruminococcus* belonging to the Lachnospiraceae and the Ruminococcaceae families become *Ruminococcus_1* and *Ruminococcus_2*, respectively. A phylo tree is created afterwards using `toTree`.

```

tax_1 <- resolveLoop(tax_tab = tax_0)
tax_tree <- toTree(data = tax_1)

# change the tree
tse_tax <- changeTree(x = tse_phy, rowTree = tax_tree,
                     rowNodeLab = rowData(tse_phy)$CONSENSUS_LINEAGE)

```

The separation of samples from different body sites appears to be worse when the data on broader resolution is used (Figure 11).

```

# aggregation data to all internal nodes
tse_agg <- aggValue(x = tse_tax,
                  rowLevel = tax_tree$node.label,
                  assay = "Count",
                  message = FALSE)

# log-transform count
assays(tse_agg)$logcounts <- log(assays(tse_agg)[[1]] + 1)

```

Specifically, we loop over each taxonomic rank and generate a t-SNE representation using data aggregated at that taxonomic rank level.

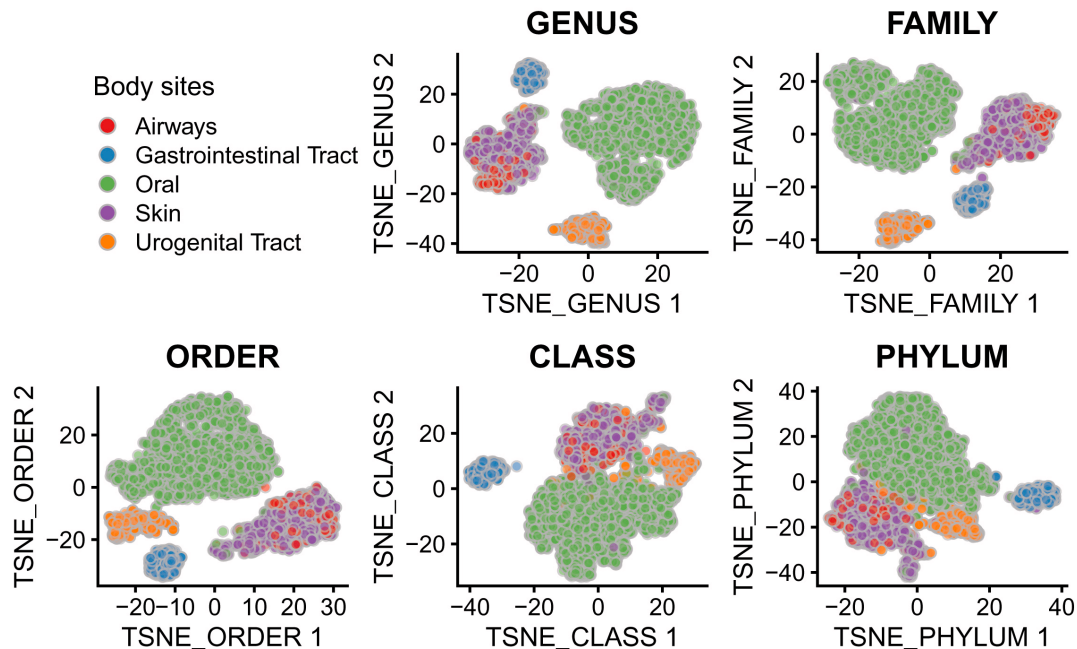


Figure 11. t-SNE plot of samples using data at different taxonomic levels. The two t-SNE components computed are plotted. Each point is a sample. Samples are colored according to the body sites.

```

tax_rank <- c("GENUS", "FAMILY", "ORDER", "CLASS", "PHYLUM")
names(tax_rank) <- tax_rank
fig_list <- lapply(tax_rank, FUN = function(x) {
  # nodes represent the specific taxonomic level
  xx <- startsWith(rowLinks(tse_agg)$nodeLab, x)

  # run t-SNE on the specific level
  xx_tse <- runTSNE(tse_agg, name = paste0("TSNE_", x),
    exprs_values = "logcounts",
    subset_row = rownames(tse_agg)[xx])

  # plot samples in the reduced dimensions
  plotReducedDim(xx_tse, dimred = paste0("TSNE_", x),
    colour_by = "HMP_BODY_SITE") +
    labs(title = x) +
    theme(plot.title = element_text(hjust = 0.5,
      size = 12)) +
    scale_fill_brewer(palette = "Set1") +
    theme(legend.position = "none") +
    guides(fill = guide_legend(override.aes = list(size=2.5)))
})

```

```

legend <- get_legend(
  # create some space to the left of the legend
  tsne_otu +
  theme(legend.box.margin = margin(0, 0, 0, 35),
    legend.position = "right")
)
plot_grid(plotlist = fig_list,
  legend, nrow = 2)

```

Summary

TreeSummarizedExperiment is an S4 class in the family of *SummarizedExperiment* classes, which enables it to work seamlessly with many other packages in Bioconductor. It integrates the *SummarizedExperiment* and the *phylo* class, facilitating data access or manipulation at different resolutions of the hierarchical structure. By providing additional functions for the *phylo* class, we support users to customize functions for the *TreeSummarizedExperiment* class in their workflows.

Data availability

Underlying data

Human Microbiome Project data (v35) was used for the presented use cases. The data can be downloaded using the R package *HMP16SData*¹⁴.

Software availability

The *TreeSummarizedExperiment* package is available at:

<https://doi.org/doi:10.18129/B9.bioc.TreeSummarizedExperiment>

Source code of the development version of the package is available at:

<https://github.com/fionarhuang/TreeSummarizedExperiment>

Archived source code as at time of publication: <http://doi.org/10.5281/zenodo.4046096>¹⁹

License: MIT

Acknowledgments

We thank Héctor Corrada Bravo, Levi Waldron, Hervé Pagès, Martin Morgan, Federico Marini, Jayaram Kancherla, Domenick Braccia, Vince Carey, Kasper D Hansen, Davide Risso, Daniel van Twisk, Marcel Ramos and other members of the Bioconductor community for their helpful suggestions.

References

- Huang R, Soneson C, Germain PL, *et al.*: **treeclimbR pinpoints the data-dependent resolution of hierarchical hypotheses.** *bioRxiv*. 2020.
[Publisher Full Text](#)
- McMurdie PJ, Holmes S: **phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data.** *PLoS One*. 2013; **8**(4): e61217.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Lun A, Risso D, Korthauer K, *et al.*: **SingleCellExperiment: S4 Classes for Single Cell Data.** 2020.
[Publisher Full Text](#)
- Morgan M, Obenchain V, Hester J, *et al.*: **SummarizedExperiment: SummarizedExperiment container.** 2020.
[Publisher Full Text](#)
- Wickham H: **Advanced r.** CRC press, second edition. 2019.
[Reference Source](#)
- Paradis E, Schliep K: **ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R.** *Bioinformatics*. 2019; **35**(3): 526–528.
[PubMed Abstract](#) | [Publisher Full Text](#)
- Rue-Albrecht K, Marini F, Soneson C, *et al.*: **iSEE: Interactive SummarizedExperiment Explorer [version 1; peer review: 3 approved].** *F1000Res*. 2018; **7**: 741.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Germain PL: **SEtools: tools for working with SummarizedExperiment.** 2020.
[Publisher Full Text](#)
- Yin T, Cook D, Lawrence M: **ggbio: an R package for extending the grammar of graphics for genomic data.** *Genome Biol.* 2012; **13**(8): R77.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Wang LG, Lam TTY, Xu S, *et al.*: **Treeio: An R Package for Phylogenetic Tree Input and Output with Richly Annotated and Associated Data.** *Mol Biol Evol.* 2019; **37**(2): 599–603.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Yu G, Smith DK, Zhu H, *et al.*: **ggtree: an r package for visualization and annotation of phylogenetic trees with their covariates and other associated data.** *Methods Ecol Evol.* 2017; **8**(1): 28–36.
[Publisher Full Text](#)
- Yu G: **tidytree: A Tidy Tool for Phylogenetic Tree Data Manipulation.** 2020.
[Reference Source](#)
- Wickham H, Chang W, Henry L, *et al.*: **ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics.** 2020.
[Reference Source](#)
- Schiffer L, Azhar R, Shepherd L, *et al.*: **HMP16SData: Efficient Access to the Human Microbiome Project Through Bioconductor.** *Am J Epidemiol.* 2019; **188**(6): 1023–1026.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Wold S, Esbensen K, Geladi P: **Principal component analysis.** *Chemometr Intell Lab.* 1987; **2**(1-3): 37–52.
[Publisher Full Text](#)
- Van Der Maaten L, Hinton G: **Visualizing Data using t-SNE.** *J Mach Learn Res.* 2008; **9**: 2579–2605.
[Reference Source](#)
- McInnes L, Healy J, Melville J: **UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.** *arXiv*. 2018.
[Reference Source](#)
- McCarthy DJ, Campbell KR, Lun AT, *et al.*: **Scater: pre-processing, quality control, normalization and visualization of single-cell RNA-seq data in R.** *Bioinformatics.* 2017; **33**(8): 1179–1186.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- HUANG R, Soneson C, Ernst F, *et al.*: **fionarhuang/TreeSummarizedExperiment: v1.4.8 TreeSummarizedExperiment (Version v1.4.8).** *Zenodo*. 2020.
<http://www.doi.org/10.5281/zenodo.4046096>

Open Peer Review

Current Peer Review Status:   

Version 1

Reviewer Report 23 November 2020

<https://doi.org/10.5256/f1000research.29440.r73185>

© 2020 Ritchie M. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Matthew E. Ritchie 

The Walter and Eliza Hall Institute of Medical Research, Parkville, Vic, Australia

Huang *et al.* describe the TreeSummarizedExperiment package, which provides well-designed S4 infrastructure that couples the phylo and SingleCellExperiment classes to create a container for high-throughput data that can be organised in a tree-like structure.

The article is structured like a vignette, providing an overview of the class (Figure 1) and stepping the reader through the process of setting up a TreeSummarizedExperiment object and accessing and assigning data to its various slots, firstly for a toy data set and then for data from the Human Microbiome Project.

The article is very clearly written, and the authors demonstrate the ability to use TreeSummarizedExperiment objects in conjunction with other established software for dealing with trees (e.g. ggtree and tidyTree) or dimensionality reduction of high-throughput data (e.g. scatter). One topic that I was interested to read more about was its potential use in a single cell RNA-seq analysis. Perhaps use cases for such applications can be added as future work. The TreeSummarizedExperiment package has been available from Bioconductor since May 2019 and it has been downloaded > 2.4K times, which indicates it is being taken up by the community.

Minor issues:

- Affiliation 5: missing 'O' in 'Oxford'.
- 'Functions operating on the phylo object.' section, sentence 2, missing word: 'such as ape [and] tidytree.'

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: Transcriptomics (bulk and single cell)

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 02 Mar 2021

Ruizhu Huang, Department of Molecular Life Sciences, University of Zurich, Zurich, Switzerland

1. **Affiliation 5: missing 'O' in 'Oxford'.**

Thank you. The typo is fixed.

2. **'Functions operating on the phylo object.' section, sentence 2, missing word: 'such as ape [and] tidytree.'**

The missing word is added now.

Competing Interests: No competing interests were disclosed.

Reviewer Report 12 November 2020

<https://doi.org/10.5256/f1000research.29440.r73184>

© 2020 Ghazanfar S. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Shila Ghazanfar

Cancer Research UK Cambridge Institute, Li Ka Shing Centre, University of Cambridge, Cambridge, UK

Huang and colleagues have written a software article presenting `TreeSummarizedExperiment` [currently version 1.6.0], a Bioconductor package aimed at providing an S4 class for omics data with hierarchical tree structure. The `TreeSummarizedExperiment` class builds on the popular `SingleCellExperiment` object class, with additional slots in which hierarchical structure, in the form of phylo objects, can be added for features (rows) and observations (columns). In addition to the object class, the package contains several functions for manipulating these objects, ranging from getting/setting/resetting the tree slots, aggregating across rows and/or columns, and various analytical tasks operating on the phylo objects.

The article is well written with clear motivation and description of the package, and addresses an important problem of performing analysis of high dimensional hierarchically structured data using object-oriented programming. I have a few further comments and questions that may improve the breadth of use of `TreeSummarizedExperiment` by the research community.

- Is there a way to simply include an argument for `aggValue()` that would swap the order to columns first and rows second, rather than requiring the user to perform two distinct operations?
- The new slots, `rowTree`, `colTree`, `rowLinks` and `colLinks` are 'getter' accessors but not currently 'setter' functions. I can imagine a popular use-case among users with an already constructed object of class `SummarizedExperiment` or `SingleCellExperiment` would be to simply use `as(, "TreeSummarizedExperiment")` and then attempt to add the additional slots, for example as the output of `hclust()`. I would suggest prioritising converting these functions to both 'getter' and 'setter', or perhaps adding a constructor usage for `TreeSummarizedExperiment` for objects that are already `SummarizedExperiment` or `SingleCellExperiment`, if possible.
- I'm interested in how `TreeSummarizedExperiment` would work in the case where the hierarchical structure is not a typical single tree, but comprising of multiple distinct tree structures. An example of such is single cell (or single clone) lineage data where there exists a tree structure within each experimental condition, but not between cells from different conditions. Would the `colTree` slot correspond to a list of trees in this case?
- How would one go about combining different `TreeSummarizedExperiment` objects? Do the typical `cbind()` and `rbind()` operations have meaning here? In which cases are they not to be used?
- I would be interested in getting to a clustered heatmap as an example of visualisation for the `TreeSummarizedExperiment`, either implemented using `ggplot2/ggtree`, or other packages like `ComplexHeatmap`?
- How would the tree structure information storage scale in terms of the number of rows and columns, or in the hierarchical structures?

Minor/cosmetic

- typo in affiliation 5.
- legend cut off in Figures 6, 7, and 8.

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: No competing interests were disclosed.

Reviewer Expertise: statistics, high throughput genomics data analysis, single cell genomics analysis, spatial gene expression analysis

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 02 Mar 2021

Ruizhu Huang, Department of Molecular Life Sciences, University of Zurich, Zurich, Switzerland

Thank you for your comments!

1. swap the order to columns first and rows second, rather than requiring the user to perform two distinct operations?

aggValue() is now deprecated and replaced by a new function, aggTSE(), that allows users to swap the order of aggregation and define different functions for the row and the column dimension.

2. The new slots, rowTree, colTree, rowLinks and colLinks are 'getter' accessors but not currently 'setter' functions. I can imagine a popular use-case among users with an already constructed object of class SummarizedExperiment or SingleCellExperiment would be to simply use as(, "TreeSummarizedExperiment") and then attempt to add the additional slots, for example as the output of hclust(). I would suggest prioritising converting these functions to both 'getter' and 'setter', or perhaps adding a constructor usage for TreeSummarizedExperiment for objects that are already

SummarizedExperiment or SingleCellExperiment, if possible.

rowTree and colTree are now both setters and getters. When the row/column tree is replaced, the rowLinks/colLinks is updated automatically. To avoid breaking links between assays and trees, we don't recommend users to modify the rowLinks/colLinks data. Therefore, rowLinks/colLinks are still kept as getters.

3. **I'm interested in how TreeSummarizedExperiment would work in the case where the hierarchical structure is not a typical single tree, but comprising of multiple distinct tree structures. An example of such is single cell (or single clone) lineage data where there exists a tree structure within each experimental condition, but not between cells from different conditions. Would the colTree slot correspond to a list of trees in this case?**

Yes, it's possible to have a list of trees in the rowTree/colTree. In the rowLinks/colLinks, we have added a new column (whichTree) to give information about which row/column tree a row/column is mapped to. We have also added a new vignette describing how to combine multiple TSEs. (

<https://www.bioconductor.org/packages/devel/bioc/vignettes/TreeSummarizedExperiment/inst/doc/7>
)

4. **How would one go about combining different TreeSummarizedExperiment objects? Do the typical cbind() and rbind() operations have meaning here? In which cases are they not to be used?**

rbind() and cbind() are now implemented for TreeSummarizedExperiment objects. To rbind() multiple TSEs successfully, it's required that the TSEs agree in the column dimension to have the same colTree() and colLinks(). Similarly, cbind() would require TSEs to have the same rowTree() and rowLinks(). More detailed information is available in the new vignette about combining multiple TSEs. (

<https://www.bioconductor.org/packages/devel/bioc/vignettes/TreeSummarizedExperiment/inst/doc/7>
)

5. **I would be interested in getting to a clustered heatmap as an example of visualisation for the TreeSummarizedExperiment, either implemented using ggplot2/ggtree, or other packages like ComplexHeatmap?**

We have added a new use case of TSE on CyTOF data, and customized a visualization function based on *ggtree*, *ggplot2* and *ggnewscale* to plot a clustered heatmap.

6. **How would the tree structure information storage scale in terms of the number of rows and columns, or in the hierarchical structures.**

We store the tree structure as a phylo object. The size of a phylo object is quite small even for a tree with 10^6 leaves (about 90 Mb). To set up the link between rows/columns to a tree, it takes only a few seconds even for 10^6 rows to a tree with 10^6 leaves.

7. typo in affiliation 5. legend cut off in Figures 6, 7, and 8.

The typo and the legend cut off are fixed.

Competing Interests: No competing interests were disclosed.

Reviewer Report 19 October 2020

<https://doi.org/10.5256/f1000research.29440.r73188>

© 2020 Lahti L. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.



Leo Lahti 

Department of Computing, Faculty of Technology, University of Turku, Turku, Finland

This software article introduces TreeSummarizedExperiment, a S4 class for hierarchically structured data in R. This provides a very generic data structure that serves for instance the single cell and microbiome bioinformatics communities, and has already gathered remarkable attention with a growing user base. The package is mature and has been available via Bioconductor for some time already.

The rationale for developing the new software tool has been clearly explained, and sufficient examples are provided. It extends the popular SingleCellExperiment class structure by bringing in tree info on data rows and cols (based on the phylo class). The new extended class has potentially many new applications, for instance in microbiome research; concrete examples are provided. The new class combines and extends other common class structures, which is very beneficial for the overall compatibility. Many tools for manipulation and use already exist based for instance on related work on the SummarizedExperiment family of classes, phylo tree structure, and the phyloseq class.

The overall description of the software is technically sound and follows standard conventions in the R/Bioconductor community. Sufficient details have been provided to allow replication of the software development and its use by others; the documentation and examples are sufficient for getting started with and interpreting outputs of the new class for anyone who has the technical skills that are needed to utilize this work.

Major

1. Efficiency of the new method could be discussed further; does this scale up to population level cohorts that have thousands of samples are increasing hierarchical resolutions?
2. How easy it would be to incorporate further supporting information on the rows and columns, for instance on DNA/RNA sequence information?

3. The class is very generic; is the idea that this package can be used as such in (hierarchical) single-cell experiments, microbiome research, and potentially other fields that have little overlap currently? Or is this package meant to be a fundamental structure that can be further extended in more specific application domains? Some more discussion on these aspects could help to contextualize the new class.

Minor

1. "ssay_data" -> "assay_data"

Is the rationale for developing the new software tool clearly explained?

Yes

Is the description of the software tool technically sound?

Yes

Are sufficient details of the code, methods and analysis (if applicable) provided to allow replication of the software development and its use by others?

Yes

Is sufficient information provided to allow interpretation of the expected output datasets and any results generated using the tool?

Yes

Are the conclusions about the tool and its performance adequately supported by the findings presented in the article?

Yes

Competing Interests: I recently discussed possible further extensions with the authors of this work. The discussion was based on my own initiative as I am working on related topics, and at that time I did not know that they had (already) submitted this manuscript for review. I do not know the authors, and we have no ongoing collaboration.

Reviewer Expertise: Bioinformatics, open research software, R/Bioconductor, microbiome research, statistical machine learning

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 02 Mar 2021

Ruizhu Huang, Department of Molecular Life Sciences, University of Zurich, Zurich, Switzerland

Thank you for reviewing our work.

1. **Efficiency of the new method could be discussed further; does this scale up to population level cohorts that have thousands of samples are increasing**

hierarchical resolutions?

TreeSummarizedExperiment (TSE) inherits the slots of the SummarizedExperiment (SE) & SingleCellExperiment (SCE) classes, and adds new slots like rowTree, colTree, rowLinks, colLinks, referenceSeq. For operations involving the inherited slots, TSE works similarly as SE and SCE. For the new slots, the data manipulation depends on the functions that users have applied on the tree object (of class phylo). These functions might be from TSE or outside TSE. For functions from TSE, either those working on the phylo tree (e.g., findDescendant, convertNode, matTree, addLabel) or those working on TSE (e.g., rowTree, colTree, rowLinks, colLinks, changeTree), takes only seconds even for a tree with up to 100,000 nodes.

2. How easy it would be to incorporate further supporting information on the rows and columns, for instance on DNA/RNA sequence information?

TSE now has a slot referenceSeq to store the sequence information of features (rows).

3. The class is very generic: is the idea that this package can be used as such in (hierarchical) single-cell experiments, microbiome research, and potentially other fields that have little overlap currently? Or is this package meant to be a fundamental structure that can be further extended in more specific application domains? Some more discussion on these aspects could help to contextualize the new class.

Currently, there is not much overlap in the community across fields, e.g, single-cell experiments, microbiome research. But, we do see that they share similarities in data structures, and can potentially share synergies in data visualization or analysis. We provide TSE as a standalone R package like SummarizedExperiment and SingleCellExperiment, and propose it as a convenient starting point to create R packages for downstream analysis or visualization of data with tree structures. We are open to update our work or receive pull requests if new features (or slots) required in a specific field are feasible to be integrated to TreeSummarizedExperiment. For example, a new optional slot referenceSeq(), which was requested mainly for microbiome data to store RNA/DNA sequencing information, has been developed by Félix G.M. Ernst, and the PR has been accepted in TreeSummarizedExperiment.

4. " ssay data" -> "assay data"

The typo is fixed.

Competing Interests: No competing interests were disclosed.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research